Modified Vector Rotational CORDIC (MVR-CORDIC) Algorithm and Architecture

Cheng-Shing Wu and An-Yeu Wu, *Member, IEEE*

***Abstract*—The CORDIC algorithm is a well-known iterative method for the computation of vector rotation. However, the major disadvantage is its relatively slow computational speed. For applications that require forward rotation (or vector rotation) only, we propose a new scheme, the modified vector rotational CORDIC (MVR-CORDIC) algorithm, to improve the speed performance of CORDIC algorithm. The basic idea of the proposed scheme is to reduce the iteration number directly while maintaining the SQNR performance. This can be achieved by modifying the basic microrotation procedure of CORDIC algorithm. Meanwhile, three searching algorithms are suggested to find the corresponding directional and rotational sequences so as to obtain the best SQNR performance. Three SQNR performance refinement schemes are also suggested in this paper. Namely, the selective prerotation scheme, selective scaling scheme, and iteration-tradeoff scheme. They can reduce and balance the quantization errors encountered in both microrotation and scaling phases so as to further improve the overall SQNR performance. Then, by combining these three refinement schemes, we provide a systematic design flow as well as the optimization procedure in the application of MVR-CORDIC algorithm. Finally, we present two VLSI architectures for the MVR-CORDIC algorithm. It shows that by using the proposed MVR-CORDIC algorithm, we can save 50% execution time in the iterative CORDIC structure, or 50% hardware complexity in the parallel CORDIC structure compared with the conventional CORDIC scheme.**

***Index Terms*—CORDIC algorithm, DSP, MVR-CORDIC algorithm, searching algorithms, VLSI.**

I. Introduction

THE COordinate Rotational DIgital Computer (CORDIC) algorithm is a well-known iterative technique to perform various basic arithmetic operations [1]–[3]. The algorithm is very attractive for hardware implementation because it uses only elementary shift-and-add steps to perform vector rotation in a two-dimensional (2-D) plane. Hence, the CORDIC algorithm can be applied to many DSP applications where rotation-based arithmetic functions are heavily utilized, such as linear system solver [4], [5], digital lattice filter [6], [7], singular value problems [8], and the fast Fourier transformation (FFT) [9], [10].

However, the major disadvantage of the CORDIC algorithm is its slow computational speed. For iterative CORDIC structure (to be discussed in Section VIII-A), the speed performance of CORDIC operation is limited by the large iteration number, N, which is generally equal to the internal wordlength, W. At algorithmic level, one trivial solution to overcome such a problem is to reduce the iteration number directly; however, signal will be seriously distorted by the approximation and quantization noise in practical implementations [11]. At circuit level, the CORDIC operation can be accelerated by introducing redundant number representation into the internal data processing, which can eliminate the carry propagation encountered in addition/subtraction operations [8], [12].

In this paper, we propose an algorithmic-level improved scheme, which is called modified vector rotational CORDIC (MVR-CORDIC) algorithm. It is very suitable for applications that use the CORDIC algorithm in only forward rotation mode (also known as vector rotation mode),[1] i.e., the rotation angles are fixed and known in advance, such as digital lattice filter [6], [7], [13] and discrete linear transformation [10], [14]–[16]. The major feature of the aforementioned applications is that the *directional sequence*, $\mu(i)$, which controls the rotation direction of each elementary angle in the microrotation phase, can be computed in advance. By reformatting $\mu(i)$ and searching for new sequences, we can reduce the iteration number significantly, while not increase the quantization noise level. This can be achieved by modifying the basic microrotation procedure of conventional CORDIC algorithm. Then, we can improve the speed performance of the conventional CORDIC algorithm.

Similar work has been reported by Hu and Naganathan [17]. In [17], with the aid of greedy search, the directional sequence, $\mu(i)$, of the angle rotated by CORDIC algorithm is *recoded*. However, the length of the resultant recoded sequence, which determines the iteration number in the microrotation phase, is not fixed but varied with the rotation angle. For certain cases, the length of the recoded sequence can be large and quite close to the upper bound of $N/2$, where N is the number of elementary angles [17]. Under such situation, in synchronous circuit design, the overall speed performance of the CORDIC-based arithmetic operation is therefore limited by those angles. Besides, the nonuniform feature of the iteration number is not suitable for modular design in VLSI implementation. To avoid the drawback, in our work, the design parameters are computed under a fixed iteration number. To solve the constrained optimization problem, we propose three searching algorithms for the MVR-CORDIC algorithm. They can provide tradeoff between computational complexity and signal-to-quantization-noise ratio (SQNR) performance.

Manuscript received September 1999; revised June 2001. This paper was supported in part by the National Science Council, R.O.C., under Grant NSC89-2218-E-002-108. This paper was recommended by Associate Editor M. Zaghloul.

C.-S. Wu is with the Department of Electrical Engineering, National Central University, Chung-Li 320, Taiwan, R.O.C (e-mail: benior@ee.ncu.edu.tw).

A.-Y. Wu is with the Department of Electrical Engineering, National Taiwan University, Taipei 106, Taiwan, R.O.C. (e-mail: andywu@cc.ee.ntu.edu.tw).

Publisher Item Identifier S 1057-7130(01)07497-3.

[1]The CORDIC algorithm can be operated in either forward rotation mode (vector rotational mode), or backward rotation mode (angle accumulation mode) [2], [3].

Moreover, we propose three SQNR refinement schemes for the MVR-CORDIC algorithm. The first SQNR refinement scheme introduces the concept of selective prerotation. It can carry out the microrotation phase of MVR-CORDIC algorithm with a reduced angle approximation error compared with existing approaches. The second scheme, which is employed in the scaling phase of MVR-CORDIC algorithm, is used to reduce the quantization error in the approximation of scaling factor. This can be achieved by using a selective scaling operation that combines two existing scaling techniques. The third refinement technique is called the iteration-tradeoff scheme. With this scheme, we can make tradeoff on the iteration number between the microrotation and scaling phase of the MVR-CORDIC algorithm. It can balance the quantization errors encountered in these two phases so as to further improve the overall SQNR performance.

Next, with aid of the proposed refinement schemes as well as SQNR analysis developed in the Appendix, we provide a systematic design flow and optimization procedure to facilitate the design process of MVR-CORDIC algorithm. The corresponding VLSI architectures show that we can save at least 50% execution time in the iterative CORDIC structure, and 50% hardware complexity in the parallel CORDIC structure, compared with the conventional CORDIC algorithm. Hence, low-power/high-speed CORDIC VLSI architectures become feasible.

The rest of the paper is organized as follows. In Section II, the conventional CORDIC algorithm is briefly reviewed. Then, we will discuss the strategies of MVR-CORDIC algorithm to accelerate the CORDIC rotation in vector rotational mode. In Section III, three searching algorithms are proposed to solve the problem under the constraints set by MVR-CORDIC algorithm. Then, we compare the computational complexity as well as the error performance of these three searching algorithms. In Section IV, computer simulations are performed to illustrate the relationship between error performance and design parameters. From Sections V–VII, we discuss three SQNR performance refinement schemes. The design flow is also addressed in detail. Finally, two corresponding VLSI architectures of MVR-CORDIC algorithm are derived in Section VIII, followed by the conclusions in Section IX.

II. CORDIC AND MVR-CORDIC ALGORITHM

A. CORDIC Algorithm

The CORDIC algorithm decomposes the rotation angle, θ, into predefined elementary angles, i.e.,

$$\theta = \sum_{i=0}^{N-1} \mu(i)a(i) + \xi \tag{1}$$

where

N	number of elementary angles;
$\mu(i)$	$\in \{1, -1\}$—rotation sequence, which determines the rotation angle θ;
$a(i)$	$= \arctan(2^{-i})$—ith elementary angle;
ξ	residue angle.

TABLE I
SUMMARY OF CORDIC ALGORITHM IN CIRCULAR ROTATIONAL MODE

% Initialization
Given $x(0), y(0)$, and $z(0)$
% Micro-rotation phase
FOR $i = 0$ to $N-1$

$$\begin{bmatrix} x(i+1) \\ y(i+1) \end{bmatrix} = \begin{bmatrix} 1 & -\mu(i)2^{-i} \\ \mu(i)2^{-i} & 1 \end{bmatrix} \begin{bmatrix} x(i) \\ y(i) \end{bmatrix}$$

% Angle updating
$z(i+1) = z(i) - \mu(i)a(i)$, where $a(i) = \arctan(2^{-i})$
END
% Scaling phase

$$\begin{bmatrix} x_f \\ y_f \end{bmatrix} = P \begin{bmatrix} x(N) \\ y(N) \end{bmatrix} = \frac{1}{\prod_{i=0}^{N-1}\sqrt{1+2^{-2i}}} \begin{bmatrix} x(N) \\ y(N) \end{bmatrix}$$

Based on (1), the CORDIC recurrence equations can be written as

$$\begin{aligned} x(i+1) &= x(i) - \mu(i)y(i)2^{-i} \\ y(i+1) &= y(i) + \mu(i)x(i)2^{-i} \\ z(i+1) &= z(i) - \mu(i)a(i) \end{aligned} \tag{2}$$

for $i = 0, 1, \cdots, N-1$. Due to the nature of the recurrence relation above, for data of B bits wordlength, no more than B iterations need be performed, i.e., $N \leq B$. In addition, the final values, $x(N)$ and $y(N)$, need to be scaled by an accumulated scaling factor

$$P = \frac{1}{\prod_{i=0}^{N-1}\sqrt{1+2^{-2i}}} \tag{3}$$

to retain the norm of the initial vector $[x(0), y(0)]^T$. Several CORDIC-like iteration schemes are proposed to perform the scaling operation [18]–[20]. In Table I, we summarize the basic iteration procedure of the CORDIC algorithm in the circular mode.[2] It consists of two major phases: the microrotation phase and scaling phase.

B. SQNR and Performance Indices

Before the derivation of the MVR-CORDIC algorithm, we first introduce the residue angle error ξ_m as the performance index for the rotational results. The residue angle error ξ_m is defined as the angle difference between the target angle θ and the angle that can be represented by CORDIC (or MVR-CORDIC) algorithm. That is

$$\xi_m \triangleq \theta - \sum_{i=0}^{N-1} \mu(i)a(i). \tag{4}$$

Additionally, another performance index, signal-to-quantization-noise ratio (SQNR), is also employed. The usage of SQNR can give a more straightforward view about the signal quality in practical implementations. The detailed discussion of SQNR and its relationship with ξ_m are addressed in the Appendix.

[2]There are three rotational modes in CORDIC algorithm; circular mode, hyperbolic mode, and linear mode. In this paper, we focus on circular CORDIC system only.

C. The Proposed MVR-CORDIC Algorithm

In the development of the MVR-CORDIC algorithm, we make the following modifications on the microrotation procedure of the conventional CORDIC algorithm.

- **Skip Some Microrotation Angles:**
  As opposed to conventional CORDIC, we are forced to skip some microrotations in the modified scheme. By doing so, we cannot only reduce the iteration number but also improve the error performance for certain angles. For example, we can rotate the angle of $\theta = \pi/4$ by performing the microrotation of elementary angle $\theta(0) = \arctan(2^0)$ once, and skipping all remaining microrotations. Then, the residue angle error $\xi_m = 0$. On the contrary, the conventional CORDIC has to go through all the N microrotations with sequence of $\mu = [1, -1, 1, 1, 1, \ldots]$, while the residue angle error ξ_m is $7.2 * 10^{-3}$ for $N = 8$.
- **Repeat Some Microrotation Angles:**
  In the conventional CORDIC of Table I, each microrotation angle, $a(i) = \arctan(2^{-i})$, is allowed to be used only once. However, in our modified scheme, we make it more flexible so that each microrotation can be performed repeatedly. By doing so, for a rotation angle that is K times of one microrotation angle, i.e., $\theta = K \cdot a(i)$, we can simply execute the microrotation of $a(i)$ by K times instead of performing microrotation in the conventional way. Therefore, better error performance can be obtained but with reduced iteration number. For example, we can obtain $\xi_m = 0$ for rotation angle of $\theta = \pi/2$ by simply rotating elementary angle of $\theta(0) = \arctan(2^0)$ twice,[3] whereas the conventional CORDIC uses the sequence $\mu = [1, 1, 1, 1, -1, \ldots]$ and the residue angle error is as large as $\xi_m = 7 * 10^{-3}$.
- **Confine the Number of Microrotations to R_m:**
  In Table I, all N microrotations need to be executed sequentially to complete the CORDIC rotation. To obtain the best performance, N is often set to be equal to the internal wordlength, W, which is the upper bound of N in practical implementations [11]. However, in our modified algorithm, we confine the iteration number in the microrotation phase to R_m ($R_m \ll W$). As we will see, this will lead to significant speed improvement in the CORDIC operation.

Putting all of these modifications together, we can rewrite (1) as

$$\theta = \sum_{i=0}^{R_m-1} \alpha(i)\theta(s(i)) + \xi_m \tag{5}$$

where $s(i) \in \{0, 1, \ldots, W-1\}$ is the rotational sequence that determines the microrotation angle in the ith iteration, and $\alpha(i) \in \{-1, 0, 1\}$ is the directional sequence that controls the direction of the ith microrotation of $\theta(s(i))$.

To see the effects of the above modifications, we show the constellation of reachable angles of MVR-CORDIC in Fig. 1(b).

[3]The example of $\pi/2$ is used to emphasize the impacts of repeating microrotation angles. However, in practical implementation, rotation of angle $\pi/2$ can be simply accomplished by setting $x_f = -y(0)$ and $y_f = x(0)$ without going through CORDIC rotation.

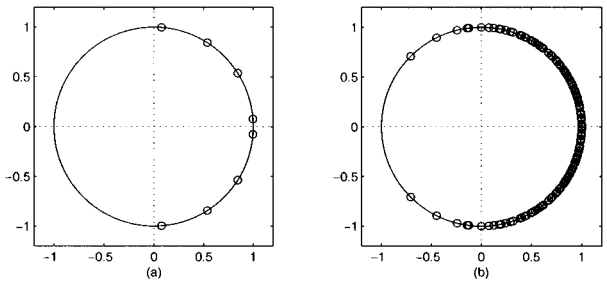

Fig. 1. Constellation of reachable angles. (a) Conventional CORDIC with $N = R_m = 3$. (b) MVR-CORDIC with $W = 4$ and $R_m = 3$.

TABLE II
SUMMARY OF THE MVR-CORDIC ALGORITHM

% Initialization
Given $x(0)$ and $y(0)$
% Micro-rotation phase
FOR $i = 0$ to $R_m - 1$

$$\begin{bmatrix} x(i+1) \\ y(i+1) \end{bmatrix} = \begin{bmatrix} 1 & -\alpha(i)2^{-s(i)} \\ \alpha(i)2^{-s(i)} & 1 \end{bmatrix} \begin{bmatrix} x(i) \\ y(i) \end{bmatrix}$$

END
% Scaling phase

$$\begin{bmatrix} x_f \\ y_f \end{bmatrix} = P \begin{bmatrix} x(N) \\ y(N) \end{bmatrix} = \frac{1}{\prod_{i=0}^{R_m-1}\sqrt{1+2^{-2s(i)}}} \begin{bmatrix} x(N) \\ y(N) \end{bmatrix}$$

The wordlength, W, is assigned to be 4, and the restricted iteration number, R_m, is 3. The reachable angles of conventional CORDIC for iteration number $N = 3$ ($N = R_m$) are also shown in Fig. 1(a) for comparison purpose. Note that the comparison is made under the condition of equal iteration number in the microrotation phase. As we can see in Fig. 1, the number of reachable angles of the proposed MVR-CORDIC is much more than the conventional CORDIC. This implies that, given the same iteration number, the MVR-CORDIC will outperform the conventional CORDIC in terms of residue angle error. Namely, given a target residue angle error, the MVR-CORDIC rotation requires fewer iterations compared with conventional approach. Consequently, the speed performance of CORDIC-based arithmetic operations can be greatly improved.

In Table II, we summarize the microrotation procedure as well as the scaling operation (to be discussed in Section VI) of the MVR-CORDIC algorithm. The main feature of the proposed MVR-CORDIC algorithm can be stated as follows.

> Given a rotation angle θ, the MVR-CORDIC attempts to accomplish the rotation in a more flexible way. It takes fewer iterations than the conventional CORDIC algorithm, while the SQNR performance is still maintained.

III. SEARCHING ALGORITHMS AND COMPARISON

A. Searching Algorithms

Consider (5) in the previous section. Now the key issue in the MVR-CORDIC algorithm is to find the best sequences of $s(i)$ and $\alpha(i)$ to minimize $|\xi_m|$, subject to the constraint that the total iteration number is confined to R_m. To solve the constrained problem, we consider the following three searching algorithms.

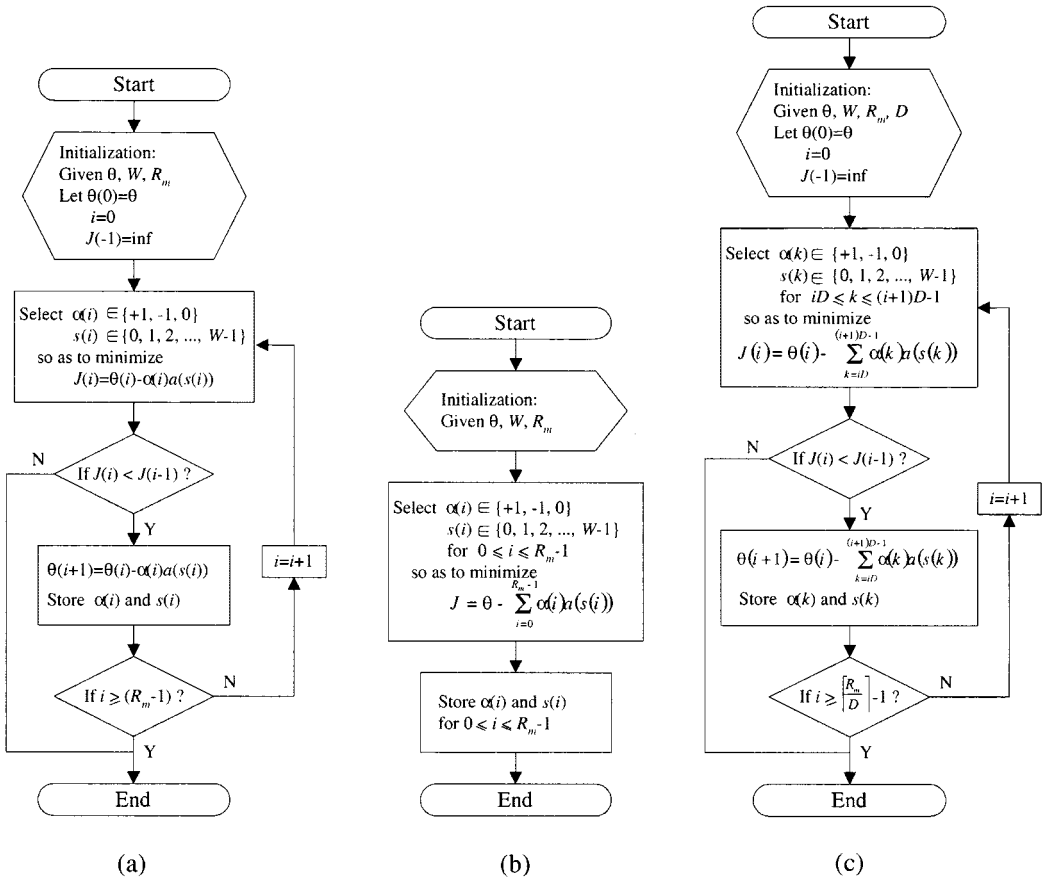

Fig. 2. The flowcharts of the proposed searching algorithms. (a) Greedy algorithm. (b) Exhaustive algorithm. (c) Semigreedy algorithm.

1) **Greedy Algorithm:**

In the greedy algorithm, we try to approach the target rotation angle, θ, step by step. That is, in each step, decisions are made on $\alpha(i)$ and $s(i)$ by choosing the best combination of $\alpha(i)a(s(i))$ so as to minimize $|\xi_m|$. Specifically, $\alpha(i)$ and $s(i)$ are determined such that the error function of $J(i) = |\theta(i) - \alpha(i)a(s(i))|$ is minimized, where $\theta(i)$ is the residue angle in ith step, defined as

$$\theta(i) = \theta - \sum_{m=0}^{i-1} \alpha(m)a(s(m)). \tag{6}$$

The searching algorithm is terminated if no further improvement can be found, i.e., $J(i) \geq J(i-1)$, or $\alpha(R_m - 1)$ and $s(R_m - 1)$ are determined at the end of the searching algorithm. The detailed flowchart of the greedy algorithm is shown in Fig. 2(a). This approach is similar to the one used in [17]. However, the major difference is that in [17], the greedy algorithm terminates only when the residue angle error cannot be further reduced. The total iteration number is not fixed.

2) **Exhaustive Algorithm:**

The second approach to solve the constrained problem is the exhaustive algorithm. The idea is to search for the entire solution space, i.e., all the possible combinations of $\sum_{i=0}^{R_m-1} \alpha(i)a(s(i))$, in one single step. Decisions for $\alpha(i)$ and $s(i)$, for $0 \leq i \leq R_m - 1$, are made such that the error function

$$J = \left| \theta - \sum_{i=0}^{R_m-1} \alpha(i)a(s(i)) \right| \tag{7}$$

is minimized. Obviously, the exhaustive searching algorithm produces global optimal solution. The flowchart of the algorithm is depicted in Fig. 2(b).

3) **Semigreedy Algorithm:**

The last searching algorithm is the semigreedy algorithm. Actually, we can treat the semigreedy algorithm as a combination of greedy and exhaustive algorithm. The whole searching space of $\alpha(i)$ and $s(i)$ for $0 \leq i \leq R_m - 1$ are divided into several sections with D iterations as a segment. We call such a segment as a block, and D is termed as the block length. The segmentation scheme is illustrated in Fig. 3. In the semigreedy algorithm, the exhaustive search is performed within each isolated block, and the connection between each consecutive blocks is determined in the greedy manner. Specifically, in the ith block (corresponds to ith step in performing the searching algorithm), the decisions of $\alpha(k)$ and $s(k)$ for

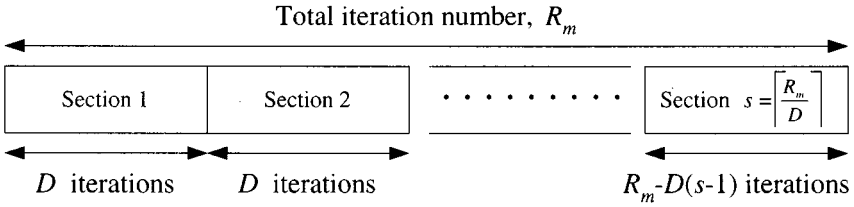

Fig. 3. The segmentation of the searching space with block length D in the semigreedy algorithm.

TABLE III
COMPUTATIONAL COMPLEXITY OF THE PROPOSED THREE SEARCHING ALGORITHMS

	Conventional CORDIC with $N = R_m$	MVR-CORDIC		
		Greedy Algorithm	Exhaustive Algorithm	Semi-greedy Algorithm
Loop Number N_L	R_m	$R_m(2N+1)$	$(2N+1)^{R_m}$	$(\frac{R_m}{D})(2N+1)^D$
Order	Constant	$O(N)$	$O(N^{R_m})$	$O(N^D)$
Run Time	Shortest	Short	Long	Middle

TABLE IV
THE COMPUTATIONAL COMPLEXITY WITH $N = 16$, $R = 6$, AND $D = 3$, AND THE ERROR PERFORMANCE COMPARISON WITH $N = 8$, $R = 4$, AND $D = 2$

	Conventional CORDIC with $N = R_m$	MVR-CORDIC		
		Greedy Algorithm	Exhaustive Algorithm	Semi-greedy Algorithm
Loop Number N_L	6	198	1,291,467,969	71,874
Normalized Loop Number	1	33	$2.2*10^8$	$1.2*10^4$
Averaged $\lvert\xi_m\rvert$	$6.289*10^{-2}$	$2.147*10^{-3}$	$8.903*10^{-4}$	$1.979*10^{-3}$

$iD \le k \le (i+1)D-1$ are made to minimize the cost function

$$J(i) = \left| \theta(i) - \sum_{k=iD}^{(i+1)D-1} \alpha(k)a(s(k)) \right| \tag{8}$$

where

$$\theta(i) = \theta - \sum_{m=0}^{i-1} \left[\sum_{k=mD}^{(m+1)D-1} \alpha(k)a(s(k)) \right] \tag{9}$$

is the residue angle in the ith step. Fig. 2(c) illustrates the detailed flowchart of the proposed semigreedy algorithm.

B. Comparison of Computational Complexity and Error Performance

Next, we compare the computational complexity and error performance for the three searching algorithms developed above. The comparison results are shown in Table III. Three parameters N, R_m, and D are the number of elementary angles, the restricted iteration number, and the searching block length of the semigreedy algorithm, respectively. The conventional CORDIC algorithm is also included here for comparison purpose. Note that the complexity is represented in terms of loop number. We use it to approximate the execution time, since the number of loops dominates the computational complexity in performing these searching algorithms.

In addition, in Table VI, we show the numerical results of the loop number by setting $N = 16$, $R_m = 6$ and $D = 3$. The Normalized Loop Number is defined as

$$\text{Normalized } N_L = \frac{\text{Loop number of the proposed searching algorithm}}{\text{Loop number of the conventional CORDIC algorithm}}. \tag{10}$$

It can be used to illustrate the complexity gaps between these searching algorithms. In the last row, we show the averaged residue angle error for $N = 8$, $R_m = 4$ and $D = 2$. The ensemble average was carried out over 65 angles from 0 to $\pi/2$ with equal space, i.e., $\theta = 0, (\pi/128), (2\pi/128), \ldots, (64\pi/128)$.

Based on the results in Tables III and IV, we can make the following observations:

- The greedy algorithm requires the least computational complexity ($N_L = 198$) among the three algorithms,

while it generates the sequences with worst error performance ($\xi_m = 2.147 * 10^{-3}$). This algorithm can be used to give the designers a quick but rough index about the performance of a specified application by using the MVR-CORDIC algorithm. It can be also applied to the situations where iterative-design is often involved, such as lattice filter design. In this case, designer may have to go through many design iterations to determine the restricted iteration number, R_m, and the wordlength, W, so as to meet the filter specification. With the greedy search, the designer can therefore choose these design parameters within short design period.

- The exhaustive algorithm consumes the longest execution time ($N_L = 1.291 * 10^9$) while resulting in the best error performance ($\xi_m = 8.903 * 10^{-4}$). This can be applied to those angles that are often employed in DSP applications, such as the twiddle factors, $W_N^{nk} = \exp(-j(2\pi nk/N))$, in FFT/IFFT. In such applications, for a given angle, we only have to perform the algorithm once to find the results of rotational sequence and directional sequence. This algorithm can be also applied to the applications where the performance is of most importance while with no restriction on execution time.
- The exhaustive algorithm can provide the best SQNR performance. However, for large N or R_m, it becomes practically impossible due to its heavy computational complexity. In such a situation, the semigreedy algorithm can be employed instead. Actually, the semigreedy algorithm plays the role in providing tradeoffs between the other two algorithms described above. The parameter D of the semigreedy algorithm is used to control the algorithm so as to generate well error performance ($\xi_m = 1.979 * 10^{-3}$) within moderate execution time ($N_L = 71, 874$). We can treat semigreedy algorithm as the generalized version for the three proposed searching algorithms, i.e., the greedy algorithm is the semigreedy algorithm with $D = 1$, and the exhaustive algorithm is the semigreedy algorithm with $D = R_m$.

IV. Relationship Between Error Performance and Design Parameters

In this section, three experiments are conducted to illustrate the relationship between error performance and the parameters used in the three searching algorithms. As with the previous experiments, the averaged residue angle error, $|\xi_m|$, is obtained based on ensemble averaging of 65 angles from 0 to $\pi/2$ with equal space.

- **Error Performance Versus Number of Elementary Angles (N):**

  In Fig. 4, the averaged residue angle error $|\xi_m|$ is plotted versus the number of elementary angles, N, for a fixed iteration number $R_m = 4$. Note that, to obtain the best error performance in the conventional CORDIC algorithm, the number of elementary angle N is set to be W. In Fig. 4, for all the searching algorithms, the error performance improves as the number of elementary angles increases.

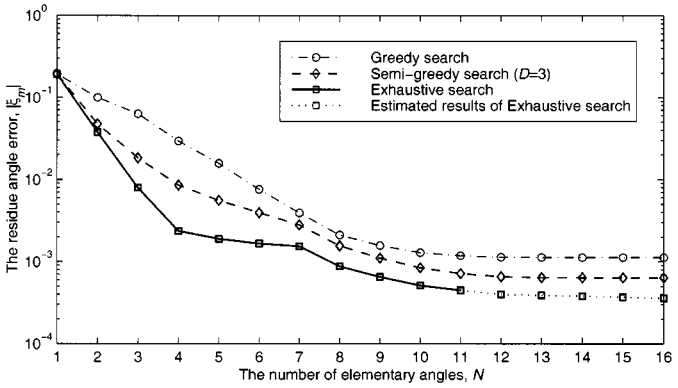

Fig. 4. Error performance versus number of elementary angles N with $R_m = 4$ and $D = 3$.

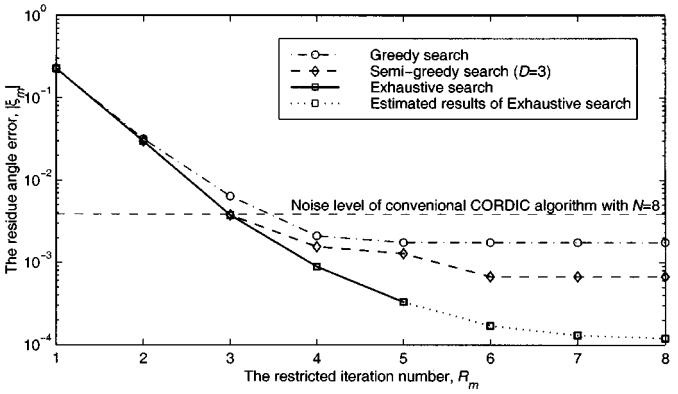

Fig. 5. Error performance versus restricted iteration number R_m with $N = 8$ and $D = 3$.

  This can be explained that we have more choices in approximating the rotation angle, θ, thus resulting in smaller residue angle error. However, the error curves are gradually saturated as N is above certain value. The reason is that the error performance cannot be improved unlimitedly only by increasing N when the restricted iteration number R_m is kept fixed. Actually, the saturation phenomenon of error performance suggests that we can perform the searching algorithms by using a smaller number of elementary angles, N' (say $N' = 11$ in this case), instead of using N directly. By doing so, we can reduce the computational complexity in running the searching algorithms while retaining compatible error performance.

- **Error Performance Versus Restricted Iteration Number (R_m):**

  Fig. 5 emphasizes on the relationship between the error performance and the restricted iteration number, R_m, for the algorithms of interest. Similar to Fig. 4, the results presented in Fig. 5 show that increasing R_m has the effect of improving error performance. Also, the error curves also saturate gradually for large R_m. The phenomenon can be explained in the similar way as with Fig. 4.

  One important observation is as follows. First, we use the horizontal dashed line to represent the averaged noise level of conventional CORDIC algorithm with $N = 8$. We can find that by using greedy search, the

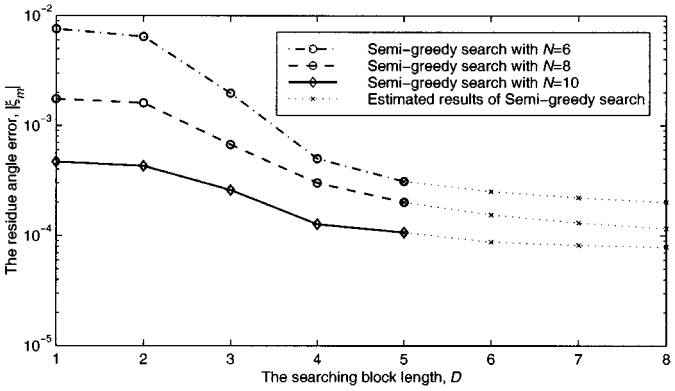

Fig. 6. Error performance versus searching block length, D, with $N = 6$, $N = 8$, $N = 10$, and $R_m = 3$.

MVR-CORDIC algorithm can perform equally well as conventional CORDIC algorithm with only 4 iterations ($R_m = 4$). For the semigreedy and the exhaustive search, even fewer iterations ($R_m = 3$ in this case) are needed to achieve compatible error performance. Recall that the conventional CORDIC algorithm has to go through 8 ($N = 8$) microrotations to reach such a noise level. In addition, computer simulations also indicates that by using semigreedy algorithm with moderate D, the MVR-CORDIC algorithm requires only $1/3$ iterations (microrotations), in an average sense, to achieve comparable (or even better) error performance compared with the conventional CORDIC algorithm.

- **Error Performance Versus Searching Block Length (D):**

  Fig. 6 depicts the residue angle error versus the searching block length (D). As can be seen, we can obtain better performance by increasing the searching block length of semigreedy algorithm. The results confirm our argument that, for larger D, the semigreedy behaves like exhaustive algorithm. On the other hand, when D is small, the essence of greedy algorithm will arise due to the confined searching space. Moreover, the saturation phenomenon suggests that we can use semigreedy algorithm with moderate value of D (say $D = 5$ in this case) to obtain a near optimum error performance without going through exhaustive search. Meanwhile, the saving in computational complexity is significant. For example, in this experiment with $N = 8$ and $D = 5$, the computational complexity rate between exhaustive search and semigreedy search is only about 0.0326%, and the performance difference is below -60 dB.

V. Selective Prerotation Scheme for MVR-CORDIC Algorithm

A. Conventional Prerotation Scheme

From (5), we can easily verify that all the reachable angles of MVR-CORDIC algorithm are confined to the range of $-R_m a(0) \le \theta \le R_m a(0)$ with the setting $\alpha(i) = \pm 1$ and $s(i) = 0$, for $0 \le i \le R_m - 1$. Hence, to perform vector rotation of arbitrary angle, i.e., $\pi \ge \theta > -\pi$, directly, the MVR-CORDIC requires at least 4 iterations ($R_m = 4$) to cover all possible rotation angles. However, the residue angle error ξ_m will be increased approximately with the value of $|\theta|$ for $\pi \ge \theta > -\pi$. This can be easily explained by observing the constellation of the reachable angles in Fig. 1(a): the distribution is sparse for $|\theta|$ of large value, which results in large residue angle error. In general, the error can be suppressed by dividing the rotation of a large θ (i.e., $\pi/2 < |\theta| \le \pi$) into two steps (assume $\pi/2 < \theta \le \pi$) [17]:

1) Prerotate the initial vector $[x(0), y(0)]^T$ by an angle of π to $[x'(0), y'(0)]^T = [-x(0), -y(0)]^T$.
2) Rotate the vector $[x'(0), y'(0)]^T$ by an angle of $(\theta - \pi)$.

In step 1), the prerotation can be easily accomplished without going through CORDIC algorithm. In step 2), we can continue to perform the MVR-CORDIC rotation of angle $(\theta - \pi)$, which is a smaller angle compared with original θ. By doing so, we can keep the rotation angle in step 2) below $\pi/2$, and hence prevent the MVR-CORDIC from rotating a large angle directly. Consequently, better error performance can be obtained.

B. Selective Prerotation Scheme

Based on above design concept, we develop an improved scheme, called the Selective Prerotation Scheme, for the MVR-CORDIC rotation of arbitrary angle. The main concept of the new scheme is that we attempt to approach the rotation angle θ in either clockwise or counterclockwise. The bidirectional rotation scheme can be achieved by introducing two different prerotation angles, where one prerotation angle is greater than θ, and the other one is smaller than θ. In general, these two types of rotation with different prerotation angles behave differently in terms of ξ_m. Then, from the alternative candidates, the one with smaller ξ_m is selected so as to perform the MVR-CORDIC rotation of θ, hence the name of the method.

To be more specific, in the proposed scheme, we first divide the MVR-CORDIC rotation into 4 groups based on the quadrant of the rotation angle θ in the complex plane. Then, in each group, two rotation types with different prerotation angles are suggested to carry out the rotation. To obtain better SQNR performance, we have to evaluate the respective ξ_m for each type. Then, choose the better one from these two candidates. Similar to the conventional prerotation scheme, two steps are required to complete each MVR-CORDIC rotation: one step for rotation of prerotation angle, and other step for rotation of remaining angle with MVR-CORDIC algorithm. We summarize the proposed alternative prerotation scheme in Table V. The proposed scheme can provide a better error performance than conventional approach without increasing any hardware complexity.

C. Design Example and Simulation

To illustrate the modified scheme developed earlier, we consider the example of $\theta = 21\pi/32$. First, the rotation angle belongs to the second group, i.e., $\pi/2 < \theta \le \pi$. According to the selective prerotation scheme in Table V, the following two types of rotation procedure may be adopted:

- Type I: Prerotate angle of $\pi/2$, followed by MVR-CORDIC rotation of $(\theta - \pi/2) = 5\pi/32$.

- Type II: Prerotate angle of π, followed by MVR-CORDIC rotation of $(\theta - \pi) = -11\pi/32$.

Here, the semigreedy algorithm (with parameters of $N = 9$, $W = 9$, $R_m = 4$, and $D = 2$) is used to search for the directional sequence, $\alpha(i)$, as well as the rotational sequence, $s(i)$, for these two angles. The results for these two rotation types are listed in Table VI, where the directional and rotational sequence are represented in the vector form as

$$\overline{\alpha} = [\alpha(0), \alpha(1), \ldots, \alpha(R_m - 1)]^T$$
$$\overline{S} = [s(0), s(1), \ldots, s(R_m - 1)]^T.$$

The corresponding SQNR values are calculated based on (A.1) with $[x(0), y(0)]^T = [1, 0]^T$.[4] It can be seen clearly that, in this case, we can obtain better error performance by using Type-I rotation. As a result, we can perform the rotation of $\theta = 21\pi/32$ with MVR-CORDIC rotation procedure of Type-I in Group-II.

Moreover, in Fig. 7, the residue angle error, ξ_m, is plotted versus 65 angles distributed from 0 to π with equal space for three rotation schemes discussed in this section; namely, the direct rotation (no prerotation phase), the conventional scheme, and the proposed modified scheme. The experiment is carried out based on the semigreedy algorithm with parameters of $N = 8$, $W = 8$, $R_m = 3$, and $D = 2$. Based on the simulation results in Fig. 7, we can make the following observations.

- Unlike the direct rotation approach, the error performance of the selective prerotation scheme will not degrade as $|\theta|$ increases.
- The selective prerotation scheme can provide apparent improvement compared with conventional prerotation scheme for $2\pi/8 < |\theta| < 6\pi/8$. The reason is that, for $2\pi/8 < |\theta| < 6\pi/8$, we can still perform the MVR-CORDIC rotation of θ that is smaller than $2\pi/8$. However, the conventional scheme has to perform the rotation with angle that is greater than $2\pi/8$.
- The proposed scheme consistently behaves best among the three schemes for all the rotation angles. The averaged residue angle error of the 65 angles for direct rotation scheme, conventional scheme, and proposed scheme are $1.2 * 10^{-1}$, $5.9 * 10^{-3}$, and $1.7 * 10^{-3}$, respectively.

VI. Selective Scaling Scheme for MVR-CORDIC Algorithm

A. Scaling Operation

In this section, we consider an implementation issue: the scaling phase of MVR-CORDIC algorithm. The use of scaling operation is intended to ensure the preservation of the norm of the vector, $[x(0), y(0)]^T$, after the sequence of microrotations. For convenience of representation, the scaling factor, P, of MVR-CORDIC algorithm in Table II is represented as [1]–[3]

$$P = \left(\prod_{i=0}^{R_m - 1} \sqrt{1 + 2^{-2s(i)}} \right)^{-1}. \tag{11}$$

[4]These SQNR values are obtained without considering the effects of scaling operation, which will be discussed in the next section.

TABLE V
Summary of the Rotation Procedure of Selective Prerotation Scheme for MVR-CORDIC Algorithm

Group of θ	Type	Pre-rotation		MVR-CORDIC Rotation Angle
		Angle	Operation	
$0 < \theta \leqslant \pi/2$	Type I	0	$\begin{bmatrix} x'(0) \\ y'(0) \end{bmatrix} = \begin{bmatrix} x(0) \\ y(0) \end{bmatrix}$	θ
	Type II	π/2	$\begin{bmatrix} x'(0) \\ y'(0) \end{bmatrix} = \begin{bmatrix} -y(0) \\ x(0) \end{bmatrix}$	θ-π/2
$\pi/2 < \theta \leqslant \pi$	Type I	π/2	$\begin{bmatrix} x'(0) \\ y'(0) \end{bmatrix} = \begin{bmatrix} -y(0) \\ x(0) \end{bmatrix}$	θ-π/2
	Type II	π	$\begin{bmatrix} x'(0) \\ y'(0) \end{bmatrix} = \begin{bmatrix} -x(0) \\ -y(0) \end{bmatrix}$	θ-π
$-\pi < \theta \leqslant -\pi/2$	Type I	-π	$\begin{bmatrix} x'(0) \\ y'(0) \end{bmatrix} = \begin{bmatrix} -x(0) \\ -y(0) \end{bmatrix}$	θ+π
	Type II	-π/2	$\begin{bmatrix} x'(0) \\ y'(0) \end{bmatrix} = \begin{bmatrix} y(0) \\ -x(0) \end{bmatrix}$	θ+π/2
$-\pi/2 < \theta \leqslant 0$	Type I	-π/2	$\begin{bmatrix} x'(0) \\ y'(0) \end{bmatrix} = \begin{bmatrix} y(0) \\ -x(0) \end{bmatrix}$	θ+π/2
	Type II	0	$\begin{bmatrix} x'(0) \\ y'(0) \end{bmatrix} = \begin{bmatrix} x(0) \\ y(0) \end{bmatrix}$	θ

TABLE VI
SQNR Results for Rotation Angle of $\theta = 21\pi/32$ for MVR-CORDIC Algorithm With the Proposed Rotation Scheme

Rotation angle, θ	Type	Rotational indicator, $\overline{S}$ Directional indicator, $\overline{\alpha}$	Residue angle error, $\lvert\xi_m\rvert$	SQNR (dB)
θ = 21π/32	Type I	$\overline{S} = [2, 2]^T$ $\overline{\alpha} = [1, 1]^T$	$9.165 * 10^{-4}$	60.76 dB
	Type II	$\overline{S} = [0, 2, 5, 6]^T$ $\overline{\alpha} = [-1, -1, -1, -1]^T$	$2.682 * 10^{-3}$	51.22 dB

To save hardware complexity, in practical implementation the scaling operation is performed by quantizing the scaling factor, P, in two forms [18]–[20], i.e.,

$$\text{Type I:} \quad \hat{P} = \sum_{j=0}^{R_s - 1} k_j \cdot 2^{-q_j} \tag{12}$$

$$\text{Type II:} \quad \hat{P} = \prod_{j=0}^{R_s - 1} (1 + k_j \cdot 2^{-q_j}) \tag{13}$$

where

$\hat{P}$ quantized value of P;
R_s restricted iteration number in the scaling phase;
k_j $\in \{1, -1\}$; and
q_j $\in \{0, 1, \ldots, W - 1\}$.

The corresponding iteration procedures for these two scaling types are as follows:

$$\text{Type I:} \quad \begin{cases} \tilde{x}(j+1) = \tilde{x}(j) + k_j \cdot 2^{-q_j} \cdot x(R_m) \\ \tilde{y}(j+1) = \tilde{y}(j) + k_j \cdot 2^{-q_j} \cdot y(R_m) \end{cases} \tag{14}$$

with $\tilde{x}(0) = 0$, and $\tilde{y}(0) = 0$,

$$\text{Type II:} \quad \begin{cases} \tilde{x}(j+1) = \tilde{x}(j) + k_j \cdot 2^{-q_j} \cdot \tilde{x}(j) \\ \tilde{y}(j+1) = \tilde{y}(j) + k_j \cdot 2^{-q_j} \cdot \tilde{y}(j) \end{cases} \tag{15}$$

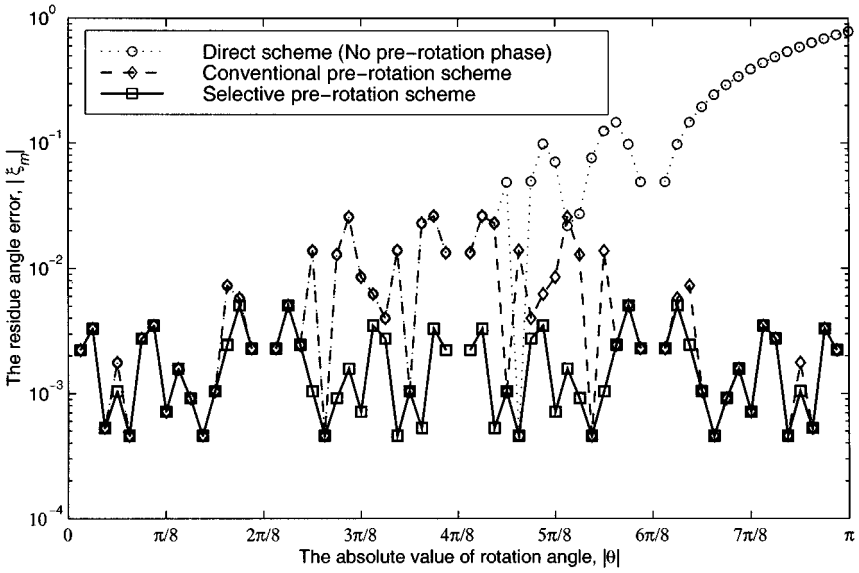

Fig. 7. SQNR performance comparison of three rotation schemes, based on semigreedy search of 65 angles uniformly distributed from 0 to π.

with $\tilde{x}(0) = x(R_m)$ and $\tilde{y}(0) = y(R_m)$. Note that $[\tilde{x}(j), \tilde{y}(j)]^T$ provide successive approximation of ideally scaled vector. It takes R_s iterations to complete the scaling phase, and the resultant vector of MVR-CORDIC is $[x_f, y_f]^T = [\tilde{x}(R_s), \tilde{y}(R_s)]^T$. By doing so, we can *approximate* the multiplication of P with only R_s shift-and-add operations, and the scaling operation can share the same circuits with the MVR-CORDIC microrotation module (to be discussed in the Section VIII), which eliminates the significant overhead of scaling multipliers.

As one can expect, this process will introduce some quantization noise, and the noise increases as R_s decreases. Similar to the microrotation phase described in Section II-B, we introduce another performance index, ξ_s, to describe the amount of error introduced by the approximation process of (12) and (13). The *scaling approximation error*, ξ_s, is defined as

$$\xi_s \triangleq \left|1 - \frac{\hat{P}}{P}\right|. \tag{16}$$

The relationship between ξ_s and SQNR performance is discussed in the Appendix.

B. Selective Scaling Scheme

In Fig. 8(a) and (b), we illustrate the distribution of the values that can be represented by (12) and (13) with $R_s = 2$ and $W = 16$. It is interesting to see that the distributions for these two types of representation are quite different, i.e., the *distribution of Type-I is dense in the region of* $\hat{P} < 0.5$; *on the contrary, the distribution of Type-II concentrated in the different region of* $\hat{P} > 0.5$. Based on the observation and apply the similar idea used in Section V-B, we are motivated to propose a novel scaling operation, called *Selective Scaling Scheme* for the MVR-CORDIC algorithm.

The basic idea of the selective scaling scheme is to combine these two types scaling operation in (12) and (13). That is, for a given scaling factor, a better strategy to quantize P is to find out the smallest ξ_s (the closest $\hat{P}$ to P) with respect to these two types of representation. We can then choose the one with smaller ξ_s from the two candidate scaling types. Hence, we can carry out the scaling operation with better SQNR performance.

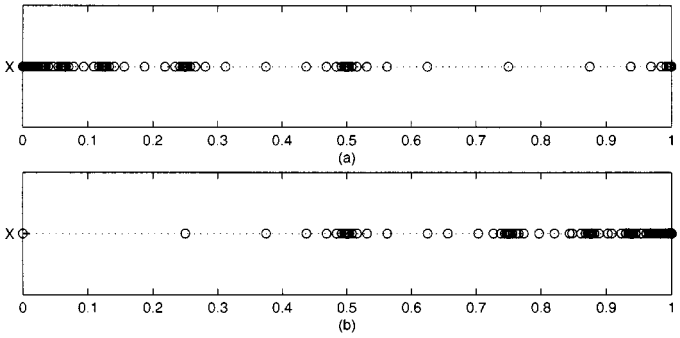

Fig. 8. The distribution of values that can be represented by (a) Scaling Type-I (b) Scaling Type-II.

C. Design Example

We use the example of $\theta = 21\pi/32$ to demonstrate the proposed scaling procedure. First, by substituting $\overline{S} = [2, 2]^T$ into (11), we obtain

$$P = \left(\sqrt{1+2^{-2\cdot 2}} \cdot \sqrt{1+2^{-2\cdot 2}}\right)^{-1} = 0.9412.$$

Assume that $R_s = 2$. We summarize the results for these two scaling types in Table VII, where k_j and q_j for $j = 0, 1, \ldots, R_s - 1$ are represented in vector form as

$$\overline{K} = [k_0, k_1, \ldots, k_{R_s-1}]^T$$
$$\overline{Q} = [q_0, q_1, \ldots, q_{R_s-1}]^T,$$

respectively. The SQNR result before performing the such a quantized scaling operation (i.e., assume floating-point scaling) is 60.76 dB. In Type-I scaling, the SQNR value drops to 47.93 dB due to the introduced quantization error of $\xi_s = 3.906 * 10^{-3}$. On the other hand, Type-II scaling has relatively low

TABLE VII
RESULTS OF THE SCALING TYPES FOR ROTATION ANGLE OF $\theta = 21\pi/32$, WHERE THE PARAMETERS ARE SET AS $R_s = 2$ AND $W = 9$

Rotation angle, $\theta = 21\pi/32$, Group-II, Rotation Type-I, with $\overline{S} = [2, 2]$, $\overline{\alpha} = [1, 1]$. SQNR = 60.76 dB (no scaling error)

Scaling factor	Scaling type	$\overline{K}$ and $\overline{Q}$	$\hat{P}$	ξ_s	SQNR (dB)
$P = 0.9412$	Type I	$\overline{K} = [1, -1]$ $\overline{Q} = [0, 4]$	0.9375	$3.906*10^{-3}$	47.93 dB
	Type II	$\overline{K} = [-1, 1]$ $\overline{Q} = [4, 8]$	0.9412	$1.526*10^{-5}$	60.76 dB

quantization noise of $\xi_s = 1.526*10^{-5}$, and the SQNR degradation is below 0.01 dB. Thus, by carefully choosing the proposed scaling scheme, we can achieve better SQNR performance of the MVR-CORDIC algorithm.

VII. ITERATION-TRADEOFF SCHEME AND DESIGN FLOW

Due to the similar nature and operations of the microrotation phase and scaling phase, one question may arise in the application of MVR-CORDIC algorithm: how to determine R_m and R_s in an optimal sense? In this section, we provide a systematic design flow to determine these two important design parameters for the MVR-CORDIC algorithm.

A. Iteration-Tradeoff Scheme for R_m and R_s

From previous discussions, we known that to carry out the complete MVR-CORDIC algorithm in Table II, we have to go through two separate phases with a total of R_T iterations. We define R_T as

$$R_T \triangleq R_m + R_s \tag{17}$$

where R_m and R_s are the restricted iteration number in microrotation phase and scaling phase, respectively. When the total iteration number is of major concern, (17) implies that we can make tradeoff between R_m and R_s. That is, we can change R_m and R_s to R'_m and R'_s, respectively, subject to the constraint

$$R'_s + R'_m = R_T. \tag{18}$$

We are justified to do so since the basic iterative operations in these two phases are almost the same. The modification can help to further improve the SQNR performance for certain rotation angles. In the following, we develop two tradeoff schemes depending on the characteristics of the rotation angle in the MVR-CORDIC algorithm.

- Case I: Trading R_s for R_m

  In this case, we attempt to gain additional rotation accuracy in the microrotation phase at the expanse of degrading precision in the scaling operation. This can be applied to the MVR-CORDIC rotation when the residue angle error, ξ_m, dominates the overall SQNR performance ($\xi_m \gg \xi_s$). It is suitable for the situation that the rotation angle lies in the region with relatively sparser distribution [see Fig. 1(a)]. Meanwhile, the corresponding scaling factor, P, can be well represented with R_s iterations. Of course, it is only meaningful that the extra SQNR gained in microrotation phase can compensate for the SQNR loss in scaling phase. An example of Case I is presented in Table VIII.

TABLE VIII
CASE I: TRADING R_s FOR R_m WITH $\theta = 7\pi/32$, $N = 16$, $D = 2$, $R_m = 3$, $R_s = 3$, $R'_m = 4$ AND $R'_s = 2$

$\theta = 7/32\ \pi$	Before Tradeoff	$R_m = 3$ $R_s = 3$	$\xi_m = 5.779*10^{-3}$ $\xi_s = 1.432*10^{-4}$	SQNR = 44.76 dB
	After Tradeoff	$R_m = 4$ $R_s = 2$	$\xi_m = 1.873*10^{-3}$ $\xi_s = 6.394*10^{-4}$	SQNR = 54.08 dB

TABLE IX
CASE II: TRADING R_m FOR R_s WITH $\theta = 8\pi/32$, $N = 16$, $D = 2$, $R_m = 3$, $R_s = 3$, $R'_m = 2$ AND $R'_s = 4$

$\theta = 8/32\ \pi$	Before Tradeoff	$R_m = 3$ $R_s = 3$	$\xi_m = 0$ $\xi_s = 1.747*10^{-3}$	SQNR = 55.16 dB
	After Tradeoff	$R_m = 2$ $R_s = 4$	$\xi_m = 0$ $\xi_s = 1.801*10^{-4}$	SQNR = 74.89 dB

- Case II: Trading R_m for R_s

  On the other hand, when the overall SQNR performance is dominated by the introduced quantization error in scaling operation, ξ_s ($\xi_s \gg \xi_m$). We attempt to obtain more accurate representation of the scaling factor but sacrificing the angle resolution in microrotation phase. It is suitable for the situation that the scaling factor lies in the region with relatively sparser distribution (see Fig. 8), while the rotation angle, θ, can be well represented with R_m elementary angles. An example of such a tradeoff case is presented in Table IX.

B. Design Flow for the MVR-CORDIC Algorithm

Based on the above iteration-tradeoff scheme, we derive the design flow as well as the optimization procedure to determine the optimal R_s and R_s for the MVR-CORDIC algorithm.

- **Step 1: Determine R_T.**

  In practical implementation, R_T must be determined according to the system requirement, such as speed, power consumption, silicon area, and, of course, the SQNR performance.
- **Step 2: Initialize R_m and R_s.**

  Once R_T is determined, we have to allocate R_m and R_s for microrotation and scaling phase, respectively. As a rule of thumb, we initially set $R_m = R_T/2$ and $R_s = R_T/2$ (assume R_T is an even integer). With this initial setting, the design flow is likely to converge to the optimum solution within fewest design iterations, in an average sense.
- **Step 3: Perform the Searching Algorithm with R_m and R_s.**

  With R_m and R_s, we can apply the semigreedy algorithm to compute $\alpha(i)$, $s(i)$, $k(i)$, and $q(i)$, as well as ξ_m and ξ_s in these two phases. For iterative design process, it

is better to use moderately small D (say $D = 1$, 2, or 3) to accelerate the computation.

- **Step 4: Estimate the SQNR Performance**.

  In general, the estimation of SQNR is a time-consuming process because we have to go through extensive computer simulation to obtain a reliable SQNR value. Fortunately, with the SQNR analysis developed in the Appendix, i.e.,

$$\text{SQNR(dB)} \approx 10\log_{10}\left(\frac{1}{\xi_m^2+\xi_s^2}\right) \tag{19}$$

  we can accurately estimate the SQNR value by simply substituting ξ_m and ξ_s into (19).

  Moreover, (19) indicates that *when ξ_m and ξ_s are of the same magnitude, the SQNR reaches its maximum value* due to the dependency of these two error indices. That is, decreasing ξ_m may have the effects of increasing ξ_s, and vice versa. We can use this property as the design guideline in determining the optimal R_m and R_s. Based on the observation as well as the iteration-tradeoff scheme developed earlier in this section, we are able to derive an optimization procedure, as described from Step 5 to Step 7.

- **Step 5: Apply the Iteration-Tradeoff Scheme**

  The selection of the tradeoff-type depends on the quantities of the errors, ξ_m and ξ_s, in these two phases. To be specific, modifications are made on R_m and R_s as

$$\begin{cases} R'_m = R_m + 1 \\ R'_s = R_s - 1 \end{cases} \text{if } \xi_m > \xi_s >, \quad \text{or} \quad \begin{cases} R'_m = R_m - 1 \\ R'_s = R_s + 1 \end{cases} \text{if } \xi_m \le \xi_s. \tag{20}$$

- **Step 6: Perform the Semigreedy Algorithm with R'_s and R'_m.**
- **Step 7: Check if SQNR is Improved**.

  By using (19), we can check if any SQNR improvement can be obtained with the modified R'_m and R'_s. If yes, accept the modification and go back to Step 5 to further improve the SQNR performance. If not, the optimization process is terminated.

- **Step 8: Perform the Semigreedy Algorithm with Large Value of D.**

  The final step of the design procedure is to perform the semigreedy algorithm with larger value of D as well as the optimized R_m and R_s, which are obtained in the iteration optimization procedure.

In Fig. 9, we illustrate the corresponding design flowchart of the proposed design methodology and optimization procedure.

C. Design Example

Consider rotational angle $\theta = 0.753\,98$ in the application of MVR-CORDIC algorithm. Assume $R_T = 8$ and the wordlength is $W = 12$. We initially set $R_m = R_s = 4$, and apply the semigreedy algorithm with small D ($D = 2$ in this case). Based on the SQNR refinement schemes developed in Sections V and VI, it can be found that $\xi_m = 7.012 * 10^{-6}$ and $\xi_s = 2.842 * 10^{-4}$ for Type-I rotation and Type-II scaling operation, respectively.

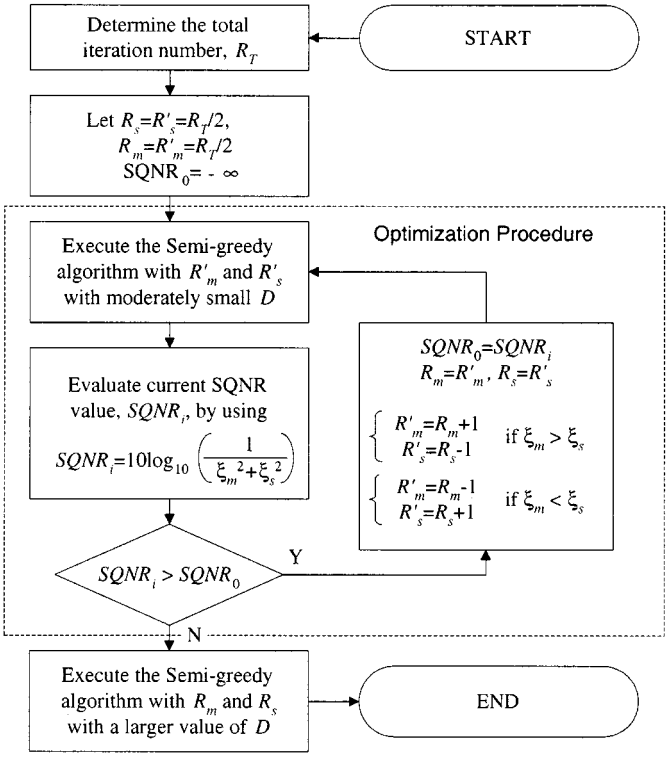

Fig. 9. The proposed design flowchart and the optimization procedure in the application of MVR-CORDIC algorithm.

The SQNR value is 70.93 dB. Next, by applying the Case-II tradeoff on R_m and R_s in (20), we can obtain an improved results of $\xi_m = 5.402 * 10^{-5}$, $\xi_s = 4.012 * 10^{-5}$, and 83.44 dB SQNR value with Type-I rotation and Type-II scaling operation. The iteration number in the microrotation and scaling phase now are $R_m = 3$ and $R_s = 5$, respectively. Since no improvement is possible, the optimization procedure is terminated. Then, we can apply the semigreedy algorithm with $R_m = 3$, $R_s = 5$, and a large value of D ($D = 4$ in this case). The resultant SQNR value can be further improved to 87.39 dB.

Fig. 10 shows all the possible combinations of R_m and R_s and their corresponding SQNR results under the constraint $R_m + R_s = R_T$. Based on the results, we can make the following observations.

1) The proposed design procedure can provide the optimal solution in the determination of R_m and R_s. That is, the resultant R_m and R_s computed by our design flow are the same as the optimal ones in Fig. 10. The important issue is that, with the aid of proposed design flow, we can obtain the optimal solution within only one design iteration, in this case, instead of checking exhaustively all possible combinations of R_m and R_s.
2) As can be seen from Fig. 10(b), the theoretical SQNR values, which are obtained by simply using (19), coincides exactly with the simulated results. The simulated SQNR values are calculated by ensemble-averaging of 10 000 output SQNR values generated by MVR-CORDIC rotation with 10 000 random input vectors $[x(0), y(0)]^T$. The results indeed confirm the validity of (19) in the fast estimation of the SQNR value.

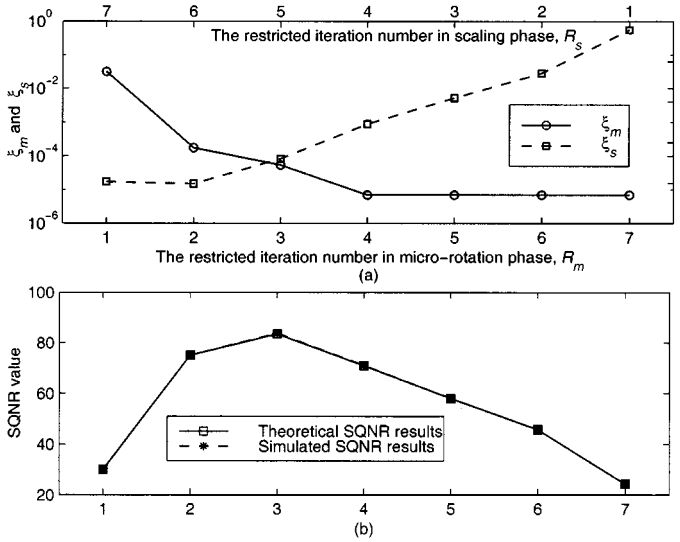

Fig. 10. (a) All possible combinations of R_m and R_s under the constraint $R_m + R_s = R_T$. (b) The corresponding theoretical and simulated SQNR values.

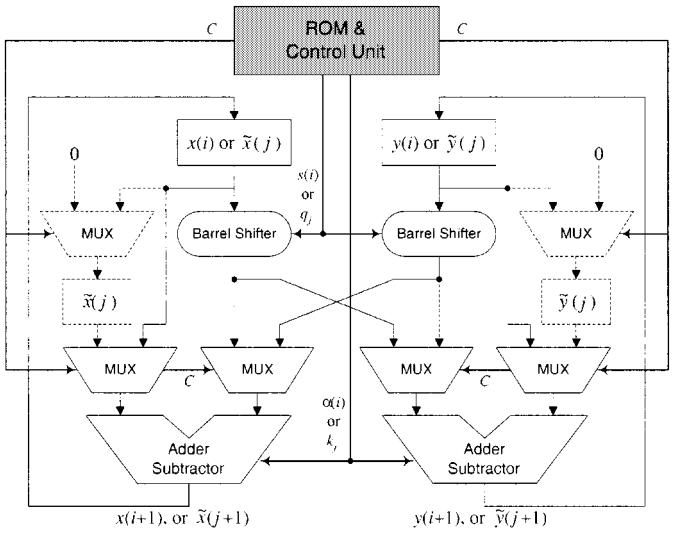

Fig. 11. The implementation of the MVR-CORDIC processor.

3) The optimal SQNR (83.44 dB) occurs under the situation that ξ_m and ξ_s are about of the same magnitude. The result confirms with our earlier argument for (19).

VIII. VLSI Implementation of MVR-CORDIC

A. Iterative MVR-CORDIC Structure

In Fig. 11, we illustrate the iterative structure for the proposed MVR-CORDIC algorithm. It is similar to the conventional iterative CORDIC structure [3]. The major difference of these two implementations lies in their control units. As shown in Fig. 11, two separate phases are performed to complete single MVR-CORDIC rotation, i.e., the microrotational phase (marked by solid line) and the scaling phase (marked by dash line). In each phase, three kinds of control signal are used to control the operations:

- $s(i)$ in microrotation phase and q_j in scaling phase: it controls the number of bits to be shifted by barrel shifters.
- $\alpha(i)$ in microrotation phase and k_j in scaling phase: it determines the operations of adder/subtracter.
- Control signal, C: it governs the phase switching of the iterative MVR-CORDIC structure and the scaling type (Type-I or Type-II) in scaling phase.

All the control signals can be generated by the proposed searching algorithm in advance, and are stored in ROM.

To evaluate the speed performance of the iterative structure, we assume that T_0 denotes the execution time to carry out single iteration of microrotation (or scaling). For MVR-CORDIC algorithm, the total execution time is $(R_m + R_s)T_0 = R_T T_0$. On the other hand, for conventional CORDIC algorithm, it takes N iterations in microrotation phase and another N' iterations in scaling phase. Thus, it requires total $(N + N')T_0$ to complete one CORDIC rotation. To make a fair comparison between these two approaches, we compare these two numbers, $(R_m + R_s)$ and $(N + N')$, under the condition of equal SQNR performance. From Section IV, we know that by using semigreedy search, the MVR-CORDIC algorithm requires an average of $R_m = N/3$ iterations to reach comparable error performance of conventional CORDIC in microrotation phase. In the scaling phase, Hu [3] has reported that $N' = N/3$ on the average. Similar result of $R_s \approx N/3$ also can be obtained for MVR-CORDIC algorithm. Hence, $R_T = 2N/3$ and $N + N' = 4N/3$. That is, we can save about 50% execution time in the iterative implementation of MVR-CORDIC algorithm compared with conventional CORDIC algorithm.

Note that the *execution time* is different from the *runtime* mentioned in Section III. They are two different design issues in the proposed MVR-CORDIC algorithm. Given a target angle θ, the runtime denotes the time to determine the design parameters of the MVR-CORDIC. The execution is defined as the hardware execution time to perform the vector rotation in VLSI circuits.

B. Parallel and Pipelined MVR-CORDIC Structure

By unfolding the iterative implementation of Fig. 11, we can obtain the parallel MVR-CORDIC structure as depicted in Fig. 12(a). The structure is composed of R_T basic MVR-CORDIC processors connected in cascade form, in which the R_m leading processors perform the microrotations and the following R_s processors execute the scaling operations. Each basic MVR-CORDIC processor performs one iteration as specified in Fig. 11. Moreover, for the case that the parallel structure is dedicated to perform a particular rotation angle, the operation of each processor is kept fixed. We can thus save the hardware complexity easily by replacing all the control circuits, barrel shifters, and multiplexers with only wire routing.

To achieve a higher data throughput rate, we can further insert pipeline stages (latches) between successive processors of parallel structure, which results in the pipelined MVR-CORDIC structure in Fig. 12(b). The pipelined structure is very suitable for real-time applications at high data bandwidth.

Due to the reduced iteration number, for parallel MVR-CORDIC structure, we can save about 50% silicon area compared with the conventional parallel CORDIC structure. The reason is that the silicon area is directly proportional to

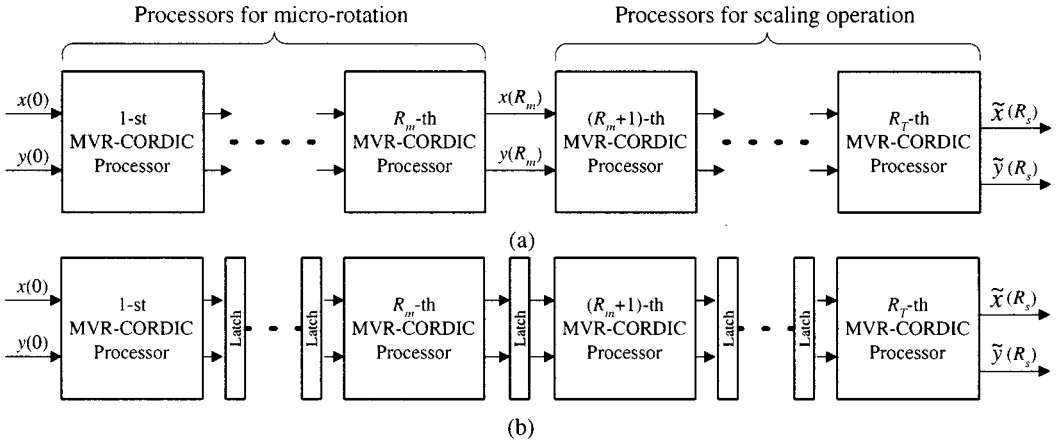

Fig. 12. (a) Parallel structure of conventional CORDIC processor array. (b) Parallel structure of MVR-CORDIC processor array.

the number of basic processors, and the numbers of processors for MVR-CORDIC and conventional CORDIC are $2N/3$ and $4N/3$, respectively. For the same reason, the critical path of parallel structure in Fig. 12(a) is only 50% compared with the conventional pipelined CORDIC structure. The latency introduced by pipelined structure can also be halved by employing the proposed MVR-CORDIC algorithm.

IX. Conclusion

In this paper, we present a new modified CORDIC algorithm, called the MVR-CORDIC algorithm, to accelerate the CORDIC operation. It can be applied to the DSP applications where rotation angles are known in advance, such as digital lattice filter and discrete orthogonal transformations. In addition, by applying the three SQNR refinements techniques developed in the paper, we can save at least 50% execution time in the iterative CORDIC structure, and 50% hardware complexity in the parallel CORDIC structure compared with the conventional CORDIC algorithm. Hence, low-power/high-speed CORDIC-based VLSI architectures for high-performance DSP applications become achievable.

Appendix
Analysis of SQNR and Its Relationship with ξ_m and ξ_s

Consider an input vector $[x(0), y(0)]^T$ and a corresponding output vector $[x_f, y_f]^T$ after the MVR-CORDIC rotation of angle θ. The SQNR [21] of such an operation is defined as

$$\text{SQNR(dB)} \triangleq 10\log_{10}\left[\frac{\text{Signal Variance}}{\text{Quantization Noise Variance}}\right] = 10\log_{10}\left[\frac{x_d^2+y_d^2}{(x_d-x_f)^2+(y_d-y_f)^2}\right] \quad \text{(A.1)}$$

where $[x_d, y_d]^T$ is the ideal output vector, given by the floating-point operation as

$$\begin{bmatrix} x_d \\ y_d \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x(0) \\ y(0) \end{bmatrix}. \quad \text{(A.2)}$$

We can identify three error sources of the quantization noise: 1) the *residue angle error*, ξ_m, in microrotation phase, 2) the *scaling approximation error*, ξ_s, in scaling phase and 3) the *rounding error* of the fixed-point arithmetic operations. In the following, we consider their impacts on the overall SQNR performance by modeling these errors as additive white noises.

- **The Noise Variance N_m Caused by Residue Angle Error, ξ_m:**

  Substituting θ in (A.2) with $\theta+\xi_m$ (or $\theta-\xi_m$), we can obtain the corresponding "ideal" output vector $[x'_d, y'_d]^T$ under the influence of ξ_m. Hence, the variance of the noise, N_m, is equal to the l_2 norm of the vector $[x_d - x'_d, y_d - y'_d]^T$; or equivalently

  $$N_m \approx (r\cdot\xi_m)^2 = r^2\cdot\xi_m^2, \qquad \text{for } \xi_m \ll 1 \quad \text{(A.3)}$$

  where r^2 is the variance of the input vector, given by $r^2 = x(0)^2+y(0)^2$ or $r^2 = x_d^2+y_d^2$. That is, N_m can be well approximated by the squared arc-length spread by residue angle ξ_m and radius r.

- **The Noise Variance N_s Caused by Scaling Approximation Error, ξ_s:**

  To express N_m in terms of ξ_s, we define the *ideally-scaled vector*, $[x_s, y_s]^T$, and the *quantized-scaled vector*, $[x'_s, y'_s]^T$, as

  $$\begin{bmatrix} x_s \\ y_s \end{bmatrix} = P\cdot\begin{bmatrix} x(R_m) \\ y(R_m) \end{bmatrix}, \quad \text{and} \quad \begin{bmatrix} x'_s \\ y'_s \end{bmatrix} = \hat{P}\cdot\begin{bmatrix} x(R_m) \\ y(R_m) \end{bmatrix} \quad \text{(A.4)}$$

  respectively. By definition, we can model the noise variance N_s as follows

  $$\begin{aligned} N_s &= (x_s-x'_s)^2+(y_s-y'_s)^2 \\ &= [x^2(R_m)+y^2(R_m)]\cdot(P-\hat{P})^2 \\ &= \left(\frac{r}{P}\right)^2\cdot(P-\hat{P})^2 = r^2\cdot\left(1-\frac{\hat{P}}{P}\right)^2 = r^2\cdot\xi_s^2 \end{aligned} \quad \text{(A.5)}$$

- **The Noise Variance N_r Caused by Arithmetic Rounding Errors:**

  Given internal wordlength, W, we can model the rounding errors introduced by arithmetic operator as a white noise with zero mean and variance of $2^{-2W}/12$ [21, Chapter 6]. As can be seen in Section VII-A, it takes R_T iterations to complete the MVR-CORDIC algorithm.

Thus, $(R_T - 1)$ add/subtract operations are performed on the intermediate $x(i)$ and $y(i)$. Accordingly, the variance of the overall rounding noise of the output vector can be simply modeled as

$$N_r = 2 \cdot (R_T - 1) \cdot \left(\frac{2^{-2W}}{12} \right). \tag{A.6}$$

Assume that all the noise sources are independent of each other, the variance of the combined noise sources of the output vector after MVR-CORDIC rotation can be written as

$$N_T \triangleq N_m + N_s + N_r. \tag{A.7}$$

In most applications, the first two noise terms dominate the overall quantization noise due to the fact that $N_m + N_s$ is several magnitude orders greater than N_r. Putting all of these together, we can relate the SQNR to the performance indicators, ξ_m and ξ_s, as

$$\begin{aligned} \text{SQNR(dB)} &\approx 10 \log_{10} \left(\frac{r^2}{r^2 \cdot \xi_m^2 + r^2 \cdot \xi_s^2} \right) \\ &= 10 \log_{10} \left(\frac{1}{\xi_m^2 + \xi_s^2} \right). \end{aligned} \tag{A.8}$$

Equation (A.8) plays an important role in determining optimal iteration numbers of R_s and R_m in Section VII-A.

REFERENCES

[1] J. E. Volder, "The CORDIC trigonometric computing technique," *IRE Trans. Electron. Computers*, vol. C-8, pp. 330–334, Sept. 1959.

[2] J. S. Walther, "A unified algorithm for elementary functions," in *Spring Joint Comp. Conf.*, 1971, pp. 379–385.

[3] Y. H. Hu, "CORDIC-based VLSI architectures for digital signal processing," *IEEE Signal Processing Mag.*, pp. 16–35, July 1992.

[4] K. Jainandunsing and E. F. Deprettere, "A new class of parallel algorithm for solving systems of linear equation," *SIAM J. Sci. Stat. Comput.*, vol. 10, pp. 880–912, Sept. 1989.

[5] Y. H. Hu and H. M. Chern, "VLSI CORDIC array structure implementation of Toeplitz eigensystem solvers," in *Proc. IEEE Int. Conf. Acoust. Speech, Signal Processing*, NM, 1990, pp. 1575–1578.

[6] P. P. Vaidyanathan, "A unified approach to orthogonal digital filters and wave digital filters based on the LBR two-pair extraction," *IEEE Trans. Circuits Syst.*, vol. CAS-32, pp. 673–686, July 1985.

[7] A. Y. Wu, K. J. R. Liu, and A. Raghupathy, "System architecture of an adaptive reconfigurable DSP computing engine," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, pp. 54–73, Feb. 1998.

[8] M. D. Ercegovac and T. Lang, "Redundant and on-line CORDIC: Application to matrix triangularization and SVD," *IEEE Trans. Computers*, vol. 39, pp. 725–740, June 1990.

[9] A. M. Despain, "Fourier transform computers using CORDIC iterations," *IEEE Trans. Computers*, vol. 23, pp. 993–1001, Oct. 1974.

[10] ——, "Very fast Fourier transform algorithms for hardware implementation," *IEEE Trans. Computers*, vol. 28, pp. 333–341, May 1979.

[11] Y. H. Hu, "The quantization effects of the CORDIC algorithm," *IEEE Trans. Signal Processing*, vol. 40, pp. 834–844, Apr. 1992.

[12] N. Takagi, T. Asada, and S. Yajima, "Redundant CORDIC methods with a constant scale factor for sine and cosine computation," *IEEE Trans. Computers*, vol. 40, pp. 989–995, Sept. 1991.

[13] E. F. Deprettere, P. Dewilde, and R. Udo, "Pipelined CORDIC architectures for fast VLSI filtering," in *Proc. IEEE Int. Conf. ASSP*, 1984, pp. 1–4.

[14] L. W. Chang and S. W. Lee, "Systolic arrays for the discrete Hartley transform," *IEEE Trans. Signal Processing*, vol. 29, pp. 2411–2418, Nov. 1991.

[15] W. H. Chen, C. H. Smith, and S. C. Fralick, "A fast computational algorithm for the discrete cosine transform," *IEEE Trans. Commun.*, vol. COM-25, pp. 1004–1009, Sept. 1977.

[16] Y. H. Hu and S. Naganathan, "Efficient implementation of the Chirp Z-transform using a CORDIC processor," *IEEE Trans. Acoust. Speech, Signal Processing*, vol. 38, pp. 352–354, Feb. 1990.

[17] ——, "An angle recoding method for CORDIC algorithm implementation," *IEEE Trans. Computers*, vol. 42, pp. 99–102, Jan. 1993.

[18] G. L. Haviland and A. A. Tuszynski, "A CORDIC arithmetic processor chip," *IEEE Trans. Computers*, vol. 29, pp. 68–79, 1980.

[19] H. M. Ahmed, J. M. Delosme, and M. Morf, "Highly concurrent computing structures for matrix arithmetic and signal processing," *IEEE Computer*, vol. 15, no. 1, pp. 65–82, 1982.

[20] J. M. Delosme, "A processor for two-dimensional symmetric eigenvalue and singular value arrays," in *Proc. 21st Asilomar Conf. Circuits, Systems, Computers*, 1987, pp. 217–221.

[21] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1989.

Cheng-Shing Wu was born in Taiwan, R.O.C., in 1973. He received the B.S. and M.S. degrees in electrical engineering in 1996 and 1997, respectively, from National Central University, Taiwan, R.O.C., where he is currently working toward the Ph.D. degree in the field of digital signal processing (DSP).

His research interests are in the areas of VLSI implementation of DSP algorithms, adaptive digital filters, and digital communication systems.

An-Yeu Wu (S'91–M'96) received the B.S. degree from National Taiwan University in 1987 and the M.S. and Ph.D. degrees from the University of Maryland, College Park, in 1992 and 1995, respectively, all in electrical engineering.

During 1987–1989, he served as a signal officer in the Army, Taipei, Taiwan, R.O.C., for mandatory military service. During 1990–1995, he was a graduate teaching assistant with the Department of Electrical Engineering, University of Maryland, College Park. From August 1995 to July 1996, he was a Member of Technical Staff at AT&T Bell Laboratories, Murray Hill, NJ, working on high-speed transmission IC designs. From 1996 to July 2000, he was with the Electrical engineering Department, National Central University. He is currently an Associate Professor with the Electrical Engineering Department of National Taiwan University, taiwan, R.O.C. His research interests include low-power/high-performance VLSI architectures for DSP and communication applications, adaptive signal processing, and multirate signal processing.